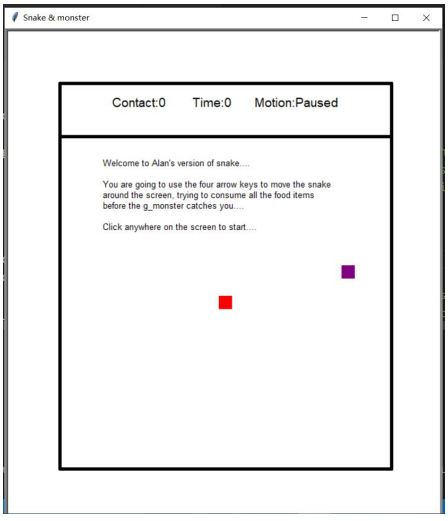
Design Doc

Overview

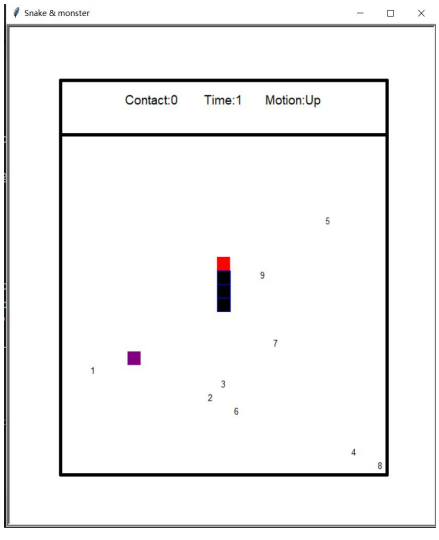
This program is an interactive snake game. In this game, the player will control the direction of the snake, trying to consume all the food on the screen. The food will be represented by digit 1 to 9 on the screen. When the snake reaches the location of a number, it will eat the food and the number will disappear. After the snake consume the food, its body will be lengthened. How long it will be lengthened depends on what number it has consumed. For instance, when the snake consumes number 4, its body will be lengthened by 4 units. The original length of the snake is 5 units. The task for the player is to consume all the food without being caught by the monster.

There will be a monster in the game who tries to approach the snake's head. Once the monster makes a head-on collision with the snake, the player will lose the game. Therefore, the player should design a safe route for the snake to consume all the food in advance. There is a contact counter in the game which count the number of times that the monster overlaps the snake's body. The player should also try to avoid the contact between the snake's body and the monster. As for the moving speed of the two characters in the game, the snake moves at a constant rate when it's not consuming any food and lengthen its body. However, once it eats any food, its moving rate would become slower. The monster moves at a rate that is randomly changed. The monster's rate would be slightly higher or lower than the snake's rate.

Once the program is executed, the initial interface will be shown. In the initial interface, there will be initial information of the status and an introduction about the game. To enter the gaming interface, the player needs to click inside the screen. Then, the monster starts to move and the timer and counter starts working. The player can use the four arrow keys to control the snake moving in four directions. The player can also press the space key to pause the game.



Initial state



Gaming state

DATA MODEL

The snake and its tails consist of turtles in square shape. Therefore, I can get the locations of the snake's head at any place. Given the head's coordinates, I can check whether the monster hit the snake's head as well as whether the snake has reached the boundaries. I use a list to store all the information about the locations of its body. This list is composed of tuples with x coordinates and y coordinates of each square turtle which represents the snake's body. The usage of the list is to check whether the monster has overlapped with the snake's body. Another list I create is the list that stores the id of each turtle of the body. This is because when the snake is extending its body, the stamp () method will provide an id for the new unit of the body. However, if the snake is just simply moving its body instead of extending, we not only need to create a new unit, but also clear the last unit. In this case, the list which store the id of each unit, follows the principle of "fast in first out" (FIFO), which is similar to the queue data structure. Thus, both the id list and the location list mentioned above will pop out the first element in the list.

As for the data model for food items, I also use turtles. Moreover, I store these turtles into a list called g_food_list. Each time when the snake's head is repositioned, I iterate through the whole list to check whether the snake's head is close enough to the locations of the turtles in the list. Another list called g_eat_list is also created for the food items. This list initially consists of nine "0", which means the all the food has not been consumed. Once the snake is close enough with the food item, the corresponding position in the g_eat_list will change its value from 0 to 1. The purpose of creating the list is to guarantee the food only be consumed once.

PROGEAM STRUCTURE

The whole program contains 29 functions. They can be mainly divided into four categories as followings:

1. Configuration

   Functions in this part mainly build up the basic elements for the game. To be specific, we need three functions to set the boundary for the whole screen, the upper status area, and the central motion area. Another two functions will be used to control the pen to write in the upper status area and the central motion area. To call these functions properly, I create one more function. Moreover, we also need to finish the configurations in keys and clicks. In total, 8 functions are needed.

2. Snake

   Snake is the main character in the game. The most basic thing in designing the game is to create a snake. Thus, we first need a function to initialize the snake. After that, we need to control the snake's motion in the game. Since the snake can move in four direction, we need four functions to control the snake's motion. The body length of the snake also needs to be updated each time when the head of the snake moves to a new location. We need a function to realize the refresh of its body length. Lastly, I write a function to detect whether the snake can move or not. In total, 7 functions are needed.

3. Monster

   The monster is another character in this game. Similar to the snake, we need to initialize the monster first. And then, we need a function to determine its moving direction, which is determined by the position of the snake's head. Lastly, we also need to set correct rate for the monster to move and also move the monster. In total, 3 functions are needed.

4. Food

   The process of configurating the food can be divided into 2 steps. Pick proper locations for the food randomly first. Then, create turtles that represent all the food and set then onto the interface. To finish these tasks, I need 2 functions.

5. Game Control

   Functions in this part due with the dynamic process of the whole game. First of all, we need to deal with the two status of the game, paused or un-paused. Then, I write two functions describing the snake's move and extend base on the logic of using stamp method. After that, I create a counter to count the number of times that the monster contacts with the snake's body. Moreover, I check whether the player lose or win and whether the snake has reached the boundaries. These three functions will be used in the last function in the snake's part. Besides, I also need to upgrade the information about status in the upper status area. Finally, to make all these functions work, we need a function to call these functions orderly such that it can control the main dynamic process of the game. Totally, I need 9 function in this part.

PROGRAM LOGIC

1. How to expend the snake?

   I extend the snake in such a way. First, I change the color of the snake's head and use the stamp function to copy a new turtle at the location of the snake's head. After that, I change the snake's head color to its original color and move the snake's head advance for 20 units.

2. How to motion the snake and the monster?

   The motion of the snake and the monster is actually base on how the snake expend. I do the same process as I do in expending the snake. In addition, I clear the units which is at the tail

of the snake's body such that the length of the body won't change.

3. How to detect the body contact between the snake and the monster?

   As the snake's body and the monster are both turtle object and global variables, I can check their location whenever they move to a new location. By measuring the difference between their x coordinates and y coordinates, we're able to know whether they contact with each other. I set the principle that they make one contact when the differences between x and y are less than 15 pixels.

FUNCTIONAL SPEC

1. draw_g_screen(), draw_g_second_margin(), draw_g_third_margin(), print_info1(), print_info2(), and setScreen():

   The first three functions create three turtles to set the basic margin of the game interface. The next two functions create two pens to write information in the upper status area and the central motion area. The last function calls the previous four functions to set up the interface.

2. Keys() & Click(x, y):

   The Keys () function configurate the key operations in the game. The Click (x, y) control the screen to let it turn to the gaming state once the player clicks the screen.

3. init_snake(), snake_up(), snake_down(), snake_right(), and snake_left():

   The init_snake() function creates the snake's head and also set some parameters. For instance, I set the size of the head to be the default value, the color to be red and the shape is square. The next four functions both control the directions of the snake.

4. Update_bodylength() & move_ablity():

   The first function updates the length of the snake and return the new length of the body. The second function check whether the snake can move. If it can, the function will return true. Otherwise, return false.

5. Three functions for monster

   The init_g_monster() function is used to initialize the monster. It creates the turtle and set the turtle. The g_monster_dir() function selects the correct direction for the monster. The g_monster_move() function makes the monster move.

6. Functions for food setting

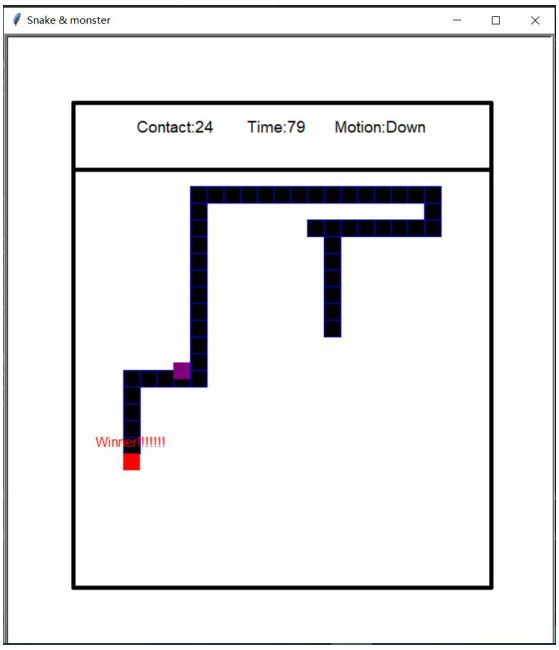
   The pick_cor() selects qualified coordinates for the food and return a dictionary that store the information about the location of each number. The set_food() function places all the food on the screen.

7. Functions for game control

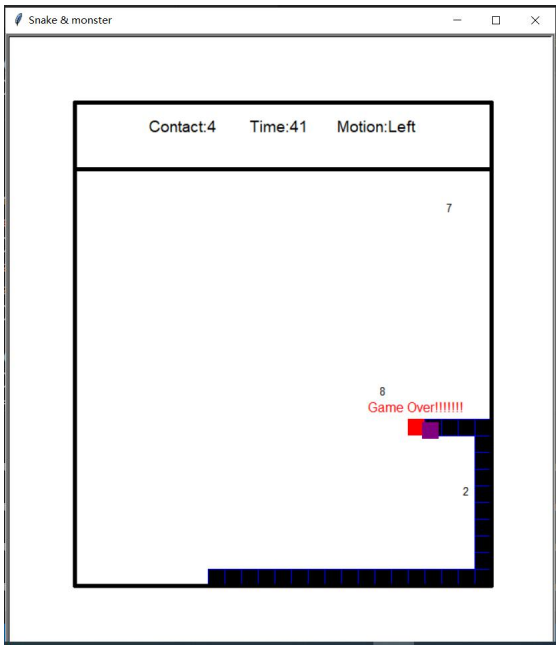
   The g_state_control() function changes the state of the game once the player presses the space key. The move() and extend() functions control the movement of the snake and monster. The contact() function works as a counter to update the number of contacts. This

function is used in the g_monster_move() function. The upgrade_g_info1() update the information in the status area. The check_game_over() function detect whether the monster overlaps with the snake. It will return a Boolean value indicating whether the player lose the game. The reach_boundary() function checks whether the snake has reached the margin of the motion area. The check_win() function checks whether the snake consume all the food without being caught. The gaming() function controls the whole game by using the functions mentioned above properly. It is the most important part of this program.

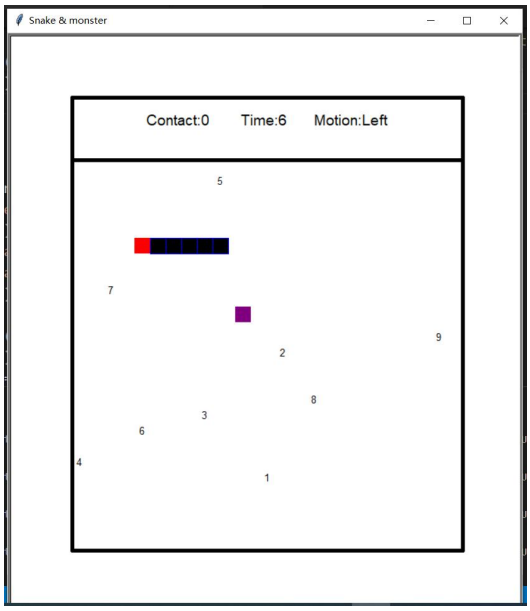
SAMPLE OUTPUT



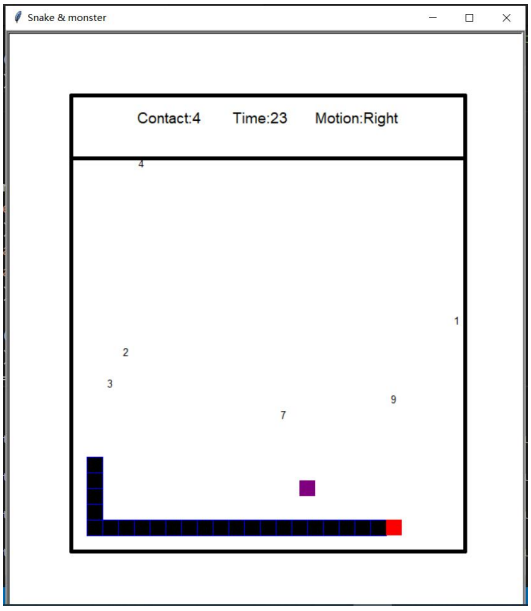
Winner!!!!!!



Game Over



With 0 food item consumed



With 3 food items consumed